Contents

1 Balltools User Manual **1**
1.1 What this is all about 1

2 Command line tools **1**
2.1 structAlign 2
2.2 secondary2csv 7
2.3 disulfideBondMaker 9
2.4 atomGeometry2csv 12
2.5 movePdb 13

3 C++ classes **14**
3.1 About the Balltools classes 14
3.1.1 Io 14
3.1.2 MoleculeHandler 14
3.1.3 DisulfideHandler 14
3.1.4 ProteinComparison 14
3.2 About the Optimization classes 15
3.2.1 Matrix 15
3.2.2 LinAlg 15
3.2.3 Derivator 15
3.2.4 Newtonizer 15
3.3 Outlook 16

1 Balltools User Manual

1.1 What this is all about

In the Course of our Metagenomics Project I developed some C++ programs and two libraries employing the APIs of BALL, CBLAS and LAPACKe and offering tools for the modification, analysis and structural alignment of tertiary structure molecules in `pdb` formatted files[1].

This manual presents the `balltools` command line tools as well as the `balltools` and `optimization` libraries as they exist today.

At the time of writing both libraries are still in the same archive but there is no real cause for that since the `optimization` classes are independent of their `balltools` siblings. This will be remedied before long.

2 Command line tools

All command line tools can be called without parameters. This will cause them to print a verbose help text explaining the available parameters and output.

[1] *http://www.pdb.org*

2.1 structAlign

Picks a *protein*0 and a *protein*1 and aligns the former to the latter one. The mathematical how-to is described in `rotationOptimization.tex`. In a nutshell it does its work in three major steps:

1. For each protein a set of weighted support vectors is determined. So far this can depend on certain secondary structure elements, the cysteine atoms in cxxc motifs, or on a preliminary 2D-Needleman-Wunsch alignment. The program will throw an error if the support vector sets are incompatible. If you intent to use the parameters `-si*` consider doing some pre-alignment analysis using `--printSecondary` and `--pretend`.

2. A preoptimization is done. First both molecules are translated such that their centers of gravity meet in the coordinate origin. Then for each pair of corresponding support vectors there exists a hyperplane of possible axes for a rotation that will, given the appropriate angle, turn one vector in such a way that it will be codirectional with the other. Usually there are many such pairs from which an overexpressed LES of normal vectors to said hyperplanes can be derived. This LES describes a constrained (solution vectors have length 1) minimization problem leading to the optimal axis for the above mentioned prior translation. This set of translation(3), angle(1) and axis(2) builds the six starting parameters θ_0 for the Newton optimization.

3. A Newton minimization is done over the six parameters describing translation and rotation of *protein*0.

```
Does a structural alignment of Protein 0 to protein 1.

Syntax: structAlign -if0 <inFname0> [-if1 <inFname1>=inFname0] \
                      [-of <outFname>=out.pdb] ...

  -if0: The input pdb file name for the pdb containing Protein 0.
    This will be the Protein which will be aligned to Protein 1.
  -if1='<inFname0>': Input pdb file name for the pdb containing Protein 1.
    This will be the Protein to which Protein 0 will be aligned to.
    If omitted both proteins will be taken from inFname0.
  -of='out.pdb': Output pdb for the aligned pt0.
  --include_pt1=0: If set to 1 the Protein 1 will also be included within the
    output pdb.
  -p0=0: Index of Protein 0 in <inFname0>.
  -p1=0: Index of Protein 1 in <inFname1>. If inFname1 was not given
    this default value is raised to 1.
  -ch={}: 0-starting index of relevant chains in Protein 0 and protein 1.
    If omitted everything will be used.
    the tag may be followed up by any number of indecies.
  -ch0={}: Cumulative specialized version of ch with a focus on pt0.
  -ch1={}: Cumulative specialized version of ch with a focus on pt1.
  --cxxc=0: Special interest out of the 'because we can' department really.
    Anyways, if added the positions of the three main atoms of the cysteines of
    each cxxc motif are added to the set of support vectors.
```

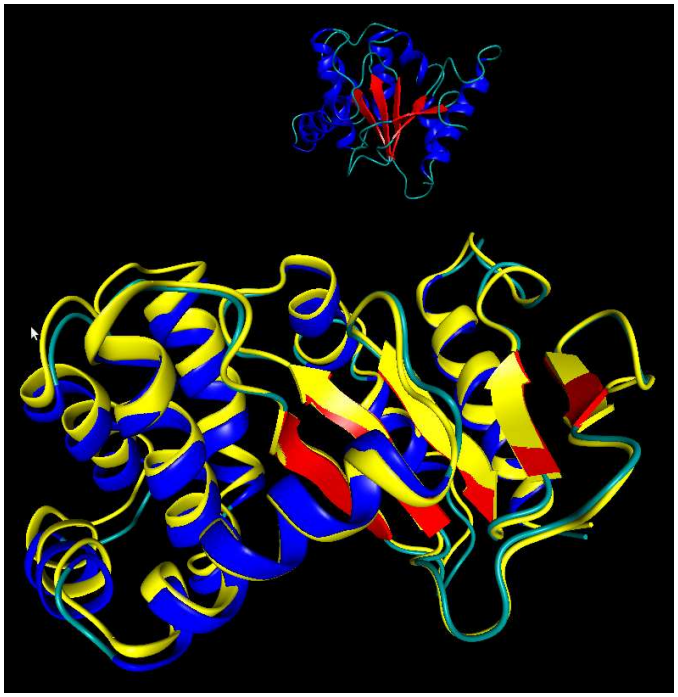
```
--needleman-wunsch, -nw=0: Build support vector set based on Needleman-Wunsch-
  alignment. To this end a Needleman-Wunsch-Alignment is done and the centers
  of gravity of aligned amino acids will be used as paired support vectors.
  The sets of relevant residues may be restricted using parameters -ri*.
--thioreductase=0: Adds to other support vector determination parameters.
  If given the six thioreductase form giving secondary structure elements of
  the proteins will be used for alignment.
--complete_secondary=0: If given it is attempted to use the complete set of
  secondary structures within the target protein.
-ri={}: 0-starting index to residues of interest within both proteins.
  If combined with --needleman-wunsch the parameters -ri* define areas of
  interest the Needleman-Wunsch alignment will be restricted to. Support
  vectors will be taken from the residues' centers of mass. For convenience
  constructs of the form '11-15' for '11 12 13 14 15' are allowed.
  The same holds true for ch, ch0, ch1, si, si0, si1, ri0 and ri1.
-ri0={}: 0-starting idx to residues of interest in protein 0.
-ri1={}: 0-starting idx to residues of interest in protein 1.
-si={}: 0-starting idx to sec. struct. elements to be used from both proteins.
-si0={}: 0-starting idx to sec. struct. elements to be used from protein 0.
-si1={}: 0-starting idx to sec. struct. elements to be used from protein 1.
-w={}: After the flag there may follow any number of positive floats. These
  will be used as weights for the Newton iteration. The nth weight is assigned
  to the nth support vector pair. When in doubt about the correct order give
  '--printSecondary -vl 4 --pretend' a shot.
  If -w is omitted uniform weighting will be used.
-mi=1000: Maximum number of iterations for the Newton optimization.
-g0=0.4: Gamma0 for the Newton iteration. If the program during an iteration
  finds a parameter gradient x it will modify the parameter list by x*g0.
-csw=1e-05: The Newton iteration will consider convergence to be achieved
  if the parameter gradient is shorter than csw for cst consecutive times.
-cst=10: If csw is underpassed cst consecutive times the program will
  call the Newton iteration converged.
--add_secondary=0: Force the addition of secondary structures using BALL
  methods. If not present and needed this option may be forced by the program.
-gop=-4: Gap opening penalty for the optional Needleman-Wunsch-Prealignment.
-gep=-1: Gap extension penalty for the optional Needleman-Wunsch-Prealignment.
--blosum=/opt/cnw/resources/blosum62.mat: Blosum Matrix File
  used for the optional Needleman-Wunsch-Prealignment.
--omitNewton=0: If set the Newton iteration will be omitted. In that case
  only the purely algebraic preoptimization will take place.
--printSecondary=0: If given the secondary structure keys for the proteins are
  printed. This may be of interest when deciding on -ri* and -si*. The amount
  of detail presented may be customized using the verbosity parameter, -vl.
--printSupport=0: If given support vector sets for the proteins are calculated
  and printed. This may be of interest when deciding on parameter -w.
--evalDiff=0: If given the weighted least squares sum of differences
  between the support vector sets before and after the alignment will be
  calculated and printed.
--rmsd: The residual root mean square positional deviation as described in
  Hasegawa, Holm 'Advances and pitfalls of protein structural alignment' after
  pairing via Needleman-Wunsch alignment. Combined with --pretend
```

```
    this will return the rmsd for the unmodified proteins.
  --pretend=0: If given no alignment will be done and no output file will be
    generated. Useful in tandem with --printSupport and --printSecondary.
  -vl=3: Verbosity level. Ranges in
    { 0:Silence, 1:Error, 2:Warning, 3:Normal, 4:Verbose }.
  --help, -h: Print this help message.

The program will return EXIT_FAILURE when the support vector sets are obviously
incompatible. If on at least ERROR verbosity (1) the program will also send
some text to STDERR.

If no support vector information is given the program will
try to use --needleman-wunsch.

Example: structAlign -if0 in0.pdb -if1 in1.pdb -of out.pdb --thioreductase \
           --printSecondary --rmsd

Markus-Hermann Koch, markus-hermann.koch@rub.de, February 2014.
```

Here is an example call for two related molecules $1A2L$ and $1A2M$.

```
./structAlign -if0 1A2L.pdb -if1 1A2M.pdb -of out.pdb --thioreductase \
  --printSecondary --printSupport
```

This produces the alignment depicted in the `BALLView` screenshot.



1A2L (red and blue) both in the background and aligned to 1A2M (which is yellow) in the foreground

In addition the call leads to a text output describing the molecules and support vectors. Note the indexing in the secondary structure blocks. These indecies can be used for the parameters `-si*`. The support vector lists have been shortened for this display.

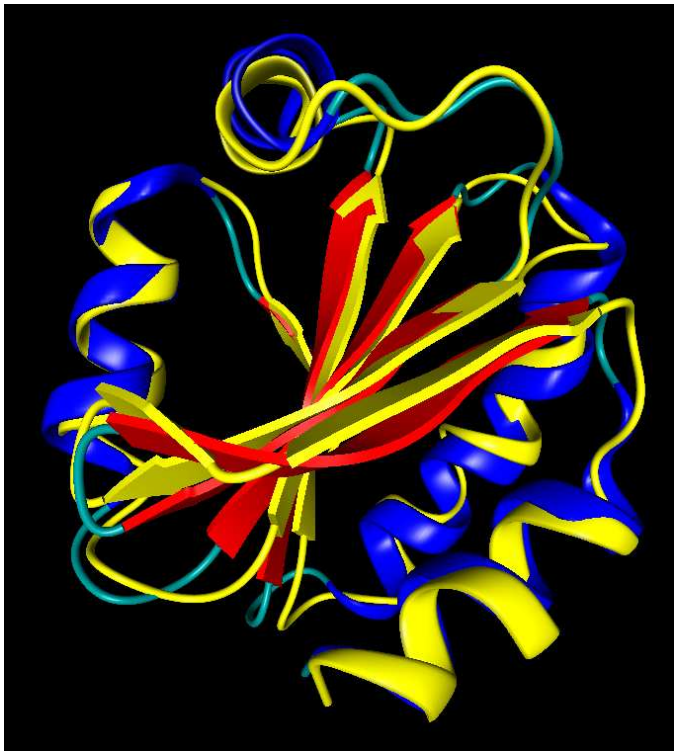
```
[..]
Support Vectors for Protein 0:
==============================
0       (56.3806 80.8755 33.4364)
1       (61.6937 79.7918 38.5236)
[..]
16      (57.4531 68.4591 29.5386)
17      (50.1681 78.0424 29.3766)
Support Vectors for Protein 1:
==============================
0       (-39.8972 62.7294 71.8136)
1       (-37.0189 63.0338 78.6559)
[..]
16      (-30.2047 58.5241 66.2088)
17      (-42.236 57.3597 61.2717)

Secondary Structure of Protein 0:
=================================
0       COIL    YEDGKQ
1       STRAND  YTT
2       COIL    LEKPVAGAPQ
3       STRAND  VLEFF
4       COIL    SFFC
5       HELIX   PHCYQFE
6       COIL    EVLH
7       HELIX   ISDNVKKK
8       COIL    LPEGVK
9       STRAND  MTKYH
10      COIL    VNFMGG
11      HELIX   DLGKDLTQAWAVAMAL
12      COIL    GV
13      HELIX   EDKVTVPLFEGVQ
14      COIL    KTQTIRS
15      HELIX   ASDIRDVFINA
16      COIL    GIK
17      HELIX   GEEYDAAW
18      COIL    NS
19      HELIX   FVVKSLVAQQEKAAAD
20      COIL    VQLRGVP
21      STRAND  AMFV
22      COIL    NGK
23      STRAND  YQL
24      COIL    NPQGMDTSN
25      HELIX   MDVFVQQYADTVKYLS
26      COIL    EK

Secondary Structure of Protein 1:
=================================
0       COIL    AQYEDGKQ
1       STRAND  YTT
2       COIL    LEKPVAGAPQ
```

```
3       STRAND  VLEFF
4       COIL    SFFC
5       HELIX   PHCYQFE
6       COIL    EVLH
7       HELIX   ISDNV
8       COIL    KKKLPEGVK
9       STRAND  MTKYH
10      COIL    VNFMGG
11      HELIX   DLGKDLTQAWAVAMA
12      COIL    LGV
13      HELIX   EDKVTVPLFEGVQ
14      COIL    KTQTIRS
15      HELIX   ASDIRDVFINA
16      COIL    GIK
17      HELIX   GEEYDAAW
18      COIL    NS
19      HELIX   FVVKSLVAQQEKAAAD
20      COIL    VQLRGVP
21      STRAND  AMFV
22      COIL    NGK
23      STRAND  YQL
24      COIL    NPQGMDTSN
25      HELIX   MDVFVQQYADTVKYLSE
26      COIL    K
[..]
```

Here is another example which uses (almost) every amino residue in the given sequences. It was created by the command[2]

```
structAlign -if0 3TRX.pdb -if1 1XWC.pdb -of 3TRX_at_1XWC_by_residues.pdb -ri 0-104
```

[2]The example is inspired by this Wikipedia example *http://en.wikipedia.org/wiki/Structural_alignment*



Alignment of 3TRX to 1XWC from a similar perspective as on said Wikipedia article.

2.2 secondary2csv

One of the first implemented tools it is heavily inspired by a bit of demo code found on the internet[3].

The program uses BALL API functionality to derive the secondary structure of the proteins in a given `pdb` file and optionally writes the results into a csv file. The help text:

```
Prints the secondary structures of the proteins within the given pdb file.
Also generates a csv formatted representation which will be written into
the output file if specified or to the console if not.

Syntax: ./secondary2csv <fname input (pdb)> [<fname output (csv)>]

Example: ./secondary2csv 1A2L.pdb

Note: If in the pdb the Amino Acid ids are not indexed consecutively and
beginning with 1 this indexing will be included into the verbose part of
the output.
```

Hence the example call

```
secondary2csv 1A2L.pdb
```

will lead to the output

[3] http://ball-trac.bioinf.uni-sb.de/wiki/CodeLibrary/ComputeSecondaryStruture

```
Adding Secondary Structures to given System.
Protein: OXIDOREDUCTASE
        Coil:     1:'3'TYR 2:'4'GLU 3:'5'ASP 4:'6'GLY 5:'7'LYS 6:'8'GLN
        Strand:   7:'9'TYR 8:'10'THR 9:'11'THR
        Coil:     10:'12'LEU 11:'13'GLU 12:'14'LYS 13:'15'PRO 14:'16'VAL 15:'17'ALA 16:'18'GLY
        Strand:   20:'22'VAL 21:'23'LEU 22:'24'GLU 23:'25'PHE 24:'26'PHE
        Coil:     25:'27'SER 26:'28'PHE 27:'29'PHE 28:'30'CYS
        Helix:    29:'31'PRO 30:'32'HIS 31:'33'CYS 32:'34'TYR 33:'35'GLN 34:'36'PHE 35:'37'GLU
        Coil:     36:'38'GLU 37:'39'VAL 38:'40'LEU 39:'41'HIS
        Helix:    40:'42'ILE 41:'43'SER 42:'44'ASP 43:'45'ASN 44:'46'VAL 45:'47'LYS 46:'48'LYS
        Coil:     48:'50'LEU 49:'51'PRO 50:'52'GLU 51:'53'GLY 52:'54'VAL 53:'55'LYS
        Strand:   54:'56'MET 55:'57'THR 56:'58'LYS 57:'59'TYR 58:'60'HIS
        Coil:     59:'61'VAL 60:'62'ASN 61:'63'PHE 62:'64'MET 63:'65'GLY 64:'66'GLY
        Helix:    65:'67'ASP 66:'68'LEU 67:'69'GLY 68:'70'LYS 69:'71'ASP 70:'72'LEU 71:'73'THR
        Coil:     81:'83'GLY 82:'84'VAL
        Helix:    83:'85'GLU 84:'86'ASP 85:'87'LYS 86:'88'VAL 87:'89'THR 88:'90'VAL 89:'91'PRO
        Coil:     96:'98'LYS 97:'99'THR 98:'100'GLN 99:'101'THR 100:'102'ILE 101:'103'ARG 102:
        Helix:    103:'105'ALA 104:'106'SER 105:'107'ASP 106:'108'ILE 107:'109'ARG 108:'110'ASI
        Coil:     114:'116'GLY 115:'117'ILE 116:'118'LYS
        Helix:    117:'119'GLY 118:'120'GLU 119:'121'GLU 120:'122'TYR 121:'123'ASP 122:'124'ALA
        Coil:     125:'127'ASN 126:'128'SER
        Helix:    127:'129'PHE 128:'130'VAL 129:'131'VAL 130:'132'LYS 131:'133'SER 132:'134'LEU
        Coil:     143:'145'VAL 144:'146'GLN 145:'147'LEU 146:'148'ARG 147:'149'GLY 148:'150'VAL
        Strand:   150:'152'ALA 151:'153'MET 152:'154'PHE 153:'155'VAL
        Coil:     154:'156'ASN 155:'157'GLY 156:'158'LYS
        Strand:   157:'159'TYR 158:'160'GLN 159:'161'LEU
        Coil:     160:'162'ASN 161:'163'PRO 162:'164'GLN 163:'165'GLY 164:'166'MET 165:'167'ASI
        Helix:    169:'171'MET 170:'172'ASP 171:'173'VAL 172:'174'PHE 173:'175'VAL 174:'176'GLI
        Coil:     185:'187'GLU 186:'188'LYS

No output file name was given. Omitting to write to disk:
COIL    1       6       YEDGKQ
STRAND  7       9       YTT
COIL    10      19      LEKPVAGAPQ
STRAND  20      24      VLEFF
COIL    25      28      SFFC
HELIX   29      35      PHCYQFE
COIL    36      39      EVLH
HELIX   40      47      ISDNVKKK
COIL    48      53      LPEGVK
STRAND  54      58      MTKYH
COIL    59      64      VNFMGG
HELIX   65      80      DLGKDLTQAWAVAMAL
COIL    81      82      GV
HELIX   83      95      EDKVTVPLFEGVQ
COIL    96      102     KTQTIRS
HELIX   103     113     ASDIRDVFINA
COIL    114     116     GIK
HELIX   117     124     GEEYDAAW
COIL    125     126     NS
HELIX   127     142     FVVKSLVAQQEKAAAD
```

```
COIL    143     149     VQLRGVP
STRAND  150     153     AMFV
COIL    154     156     NGK
STRAND  157     159     YQL
COIL    160     168     NPQGMDTSN
HELIX   169     184     MDVFVQQYADTVKYLS
COIL    185     186     EK
```

2.3 disulfideBondMaker

In the metagenomics project at our institute we had need of a function that can create or dissolve a disulfide bond within a given cxxc[4] motif. This is the primary task of this tool.

In a nutshell it works in four major steps.

1. The site of the cxxc motif of interest is either given by the user or identified automatically.
2. The disulfide bond is altered in the desired manner.
3. After the alteration BALL and Amber conjugate gradient methods are employed to do an energy minimization on the molecule thus finding the closest stable form for it.
4. The whole molecule is moved and turned in a way that will leave the cxxc motif in a defined position and orientation within the coordinate system. Thus it becomes easier to compare several molecules using tools like `BALLView`.

The console help text reads

```
Syntax: ./disulfideBondMaker ...

-if <fnameIn>: Required. Input file name.
-of <fnameOut>: Output file name. Default: "${fnameIn}.out"
-ai <index0> <index1>: Two sequence indexes (0-starting) to the cysteines to be
  used.
-cxxc: Accept only cysteines that have 2 other amino acids between them.
  Overrides -ai.
-si <skipIndex>: Sequence index (0-starting). All amino acids up to, but
  excluding the indexed one are ignored when auto-picking cysteines.
-pi <index>: Protein index. Relevant if there is more than 1 protein in
  the pdb file. Defaults to 0.
-a2b: If given the molecule will be aligned to the cystein bridge of
  interest. Else to the first cysteine (default).
-u <u0> <u1> <u2>: First vector for alignment with respect to the first
  pertinent cysteine.
  Space will be turned thus that u is parallel to (S-S) if -a2b is given or
  parallel to (S-C) else. The default [0,0,0] stands for 'skip this feature'.
  The molecule will also be translated thus that the second atom in (S-S)
  or (S-C) is moved to [0,0,0].
-v <v0> <v1> <v2>: Second vector. This will only have an effect if u is defined
```

[4] Cysteine-Wildcard-Wildcard-Cysteine

```
  and linear independent to v.
  After the u-action space will be turned around u until (S-C) if -a2b or until
  (C-C0) is within the half-plane that is spanned by alpha*u + beta*v where
  beta>=0 and alpha in R.
-mv <w0, w1, w2>: After the possible axis alignment triggered by -u and -v the
  molecule may be translated by this vector.
-as: Add secondary Structure.
-o: If given a post bridge modificational gradient optimization will be done.
  Space transformations are postponed until after this optimization.
-mi <max iterations>: Stop optimization after this many iterations.
  Auto-includes '-o'. Default value is 20000.
-mg <max gradient norm>: Maximum gradient norm for an optimization. Includes -o,
  defaults to 0.5 kJ/(mol A)
-add: Create a disulfide bridge if none is present. Auto-Drop excess hydrogen
  atoms in that case.
-rm: Remove a disulfide bridge if one is found.
-no_hc: No hydrogen correction. Per default, if a disulfide bridge is buildt or
  destroyed, hydrogen is added or removed from the respective sulphur atoms as
  appropriate. If this for some reason is undesired this parameter leads to
  skipping hydrogen correction.
-no_hyd: Do not add missing hydrogens and bonds to the entirety of the molecule.
  Default is doing so.
-h, --help: Ignores every other parameter. Print this help message and exit.

Example: ./disulfideBondMaker -if fnameIn.pdb -of fnameOut.pdb -u 0 0 1 \
  -v 0 1 0 -as -o -add -cxxc

Please note, that -o seems to require that hydrogen bonds are added.
So please refrain from using -no_hyd at the same time.

Also, please note that some Phyre-produced files already seem to contain
Cysteine Bonds. If these are met with BALL Hydrogen completion a third bond is
opened for the sulfur atoms forming the bridge. This is observable if done in
BALLView, too. A workaround for this is including -add to the call.
./disulfideBondMaker will then drop the excess hydrogen after the completion
step.

Contact: Markus-Hermann Koch, markus-hermann.koch@rub.de, October 2013
```

Here is an example call. 1A2L has no disulfide bond. The program creates it, does a standard energy minimization and moves and turns the resulting molecule in such a way that the sulphur-attached carbon C of the first cysteine comes into the origin, said sulphur S becomes codirectional with $u = [0, 0, 1]$ and the base carbon C_0 comes into the half-plane $\alpha u + \beta v$ where $\alpha \in \mathbb{R}$, $\beta \in \mathbb{R}^+$ and $v = [0, 1, 0]$.

```
disulfideBondMaker -if 1A2L.pdb \
  -of 1A2L_closed.pdb -u 0 0 1 -v 0 1 0 -as -o -add -cxxc

disulfideBondMaker -if 1A2L_closed.pdb \
  -of 1A2L_closed_corrected.pdb -u 0 0 1 -v 0 1 0 -as -o -cxxc
```

TODO: *You may wonder why there are in fact two calls above when the text mentions only one. This is to be contributed to a bug that causes the disulfide bond length to receive an unrealistic length if closed and then optimized directly.*

I have reason to believe this is due to some BALL internal reindexing which is done if an amino residue is altered (e.g. by closing a cysteine bond) which hinders Amber to calculate certain values correctly (you may notice the `AmberBend::setup` *errors within the output below)*

So far my only workaround for this is applying the tool in two consecutive steps:

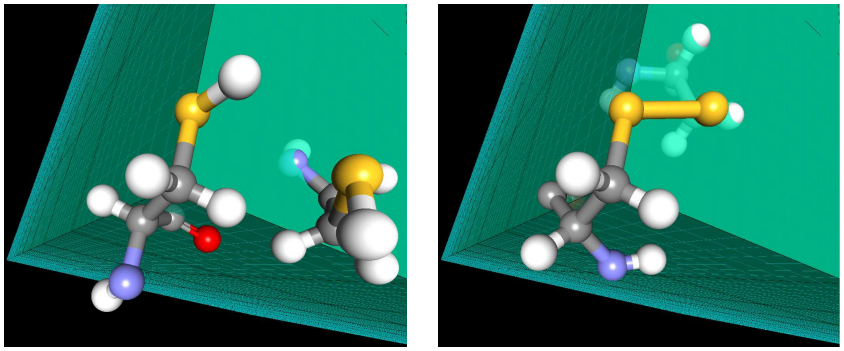
- *Load the pdb.* $\rightarrow$ *Alter the cysteine bond.* $\rightarrow$ *Save the pdb.*
- *Load the pdb.* $\rightarrow$ *Energy optimization.* $\rightarrow$ *Do translation and rotation.* $\rightarrow$ *Save the pdb.*

The mentioned TODO is to at least include this workaround into a single call to `disulfideBondMaker` *and to ultimately remove this paragraph!*

A second call merely moves 1A2L in the way described above but skips both the alteration of the disulfide bond and the subsequent optimization:

```
disulfideBondMaker -if 1A2L.pdb -of 1A2L_moved.pdb -u 0 0 1 -v 0 1 0 -as
```

The graphical representation shows a superposition of the coordinate system and the output of both molecules at the sites of the cxxc motif in question.



Cysteine bond in open and closed state oriented to both carbons and the sulphur atom within the N-terminal sided cysteine residue.

Here is the output of the first call

```
Read PDB. 186 fragments found from 1403 atoms.
Read PDB. 186 fragments found from 1403 atoms.
Adding missing hydrogens and bonds.
Found a CxxC motif at indecies 27 and 30.
Adding Secondary Structures to given System.
Deleting Hydrogen using selection code: name(HG)
Deleting Hydrogen using selection code: name(HG)
Succeeded to build bond.
Optimizing for maxIt=20000 iterations and maxGrad=0.5 kJ/(mol A).
Creating Amber Force Field.
  cannot find stretch parameters for atom types SH-SH (atoms are: CYS30:SG/CYS33:SG)
```

```
  AmberBend::setup: cannot find bend parameters for atom types:CT-SH-SH (atoms are: CYS30:CB/C
  AmberBend::setup: cannot find bend parameters for atom types:CT-SH-SH (atoms are: CYS33:CB/C
  AmberTorsion::setup: cannot find torsion parameter for:CT-SH-SH-CT (atoms are: CYS:CB/CYS:SG
Creating Conjugate Gradient Minimzer.
Setting up ConjugateGradientMinimizer. Options:
MAXIMAL_NUMBER_OF_ITERATIONS => 20000
ENERGY_OUTPUT_FREQUENCY      => 100
ENERGY_DIFFERENCE_BOUND      => 0.000100
MAX_SAME_ENERGY              => 50
MAX_GRADIENT                 => 0.500000
Given options set for Conjugate Gradient Minimizer checked and valid.
iteration 0  RMS gradient 138.603 kJ/(mol A)      total energy 52037.7 kJ/mol
iteration 100  RMS gradient 24.6961 kJ/(mol A)      total energy -3127.38 kJ/mol
iteration 200  RMS gradient 11.4432 kJ/(mol A)      total energy -14195 kJ/mol
iteration 300  RMS gradient 8.4953 kJ/(mol A)      total energy -17555.7 kJ/mol
iteration 400  RMS gradient 5.26606 kJ/(mol A)      total energy -18581.2 kJ/mol
iteration 500  RMS gradient 6.09872 kJ/(mol A)      total energy -19635.3 kJ/mol
iteration 600  RMS gradient 3.64588 kJ/(mol A)      total energy -20134.2 kJ/mol
iteration 700  RMS gradient 3.31071 kJ/(mol A)      total energy -20987.2 kJ/mol
iteration 800  RMS gradient 2.44736 kJ/(mol A)      total energy -20881.7 kJ/mol
iteration 900  RMS gradient 1.85797 kJ/(mol A)      total energy -21153.9 kJ/mol
iteration 1000  RMS gradient 1.6139 kJ/(mol A)      total energy -21241.1 kJ/mol
iteration 1100  RMS gradient 2.10721 kJ/(mol A)      total energy -21035.3 kJ/mol
iteration 1200  RMS gradient 1.20924 kJ/(mol A)      total energy -21575.2 kJ/mol
iteration 1300  RMS gradient 1.67168 kJ/(mol A)      total energy -21637.1 kJ/mol
iteration 1400  RMS gradient 1.92808 kJ/(mol A)      total energy -21699 kJ/mol
iteration 1500  RMS gradient 2.5221 kJ/(mol A)      total energy -21992.7 kJ/mol
iteration 1600  RMS gradient 2.3177 kJ/(mol A)      total energy -22414.4 kJ/mol
iteration 1700  RMS gradient 2.43856 kJ/(mol A)      total energy -22419.7 kJ/mol
iteration 1800  RMS gradient 0.739445 kJ/(mol A)      total energy -22918.5 kJ/mol
iteration 1900  RMS gradient 1.38635 kJ/(mol A)      total energy -22839.5 kJ/mol
Stopping Minimization. Final gradient norm: 0.443127 kJ / (mol Ang)
Aligning System to CYS 0. u=(0.000,0.000,1.000), v=(0.000,1.000,0.000)
```

2.4 atomGeometry2csv

Takes a protein and a list of amino residues, atoms or cxxc motifs of interest and produces a csv describing lengths and angles between the associated atoms. Due to special interest in the case of cxxc motifs and a closed disulfide bond the torsion angle of the sulphur-attached carbon atoms around the (S-S) axis defined by the disulfide bond is also included.

```
Syntax: ./atomGeometry2csv <fnameIn> [<fnameOut=${fnameIn}.csv>] ...

<fnameIn>: Required. Input file name.
[<fnameOut>]: Output file name. Default: "${fnameIn}.csv"
-pi <index>: Protein index. Relevant if there are more than 1 protein in the pdb
  file. Defaults to -1 standing for every protein in the system.
-cxxc: Special interest parameter. Include cysteine bonds from cxxc motifs.
  only for those torsion angles will be determined.
-res <residue name>: Include all atoms found in residues of this kind. -res may
  be given as parameter several times. Residue names need to be given in
```

```
  uppercase 3 letter codes.
-atom <atom name>: Include output to atoms of this name. -atom may be included.
  several times in the parameter list. Output will include the atoms of the
  given name. Atom names need to be given in BALL format like 'CYS:SG', 'SG',
  or 'Sulphur'. The first in this list is the so-called FullName which is not
  found in BALLView.
-resIndex <residueIndex>: A specific 0-starting index to a complete residue of
  interest.
-atomIndex <atomIndex>: A specific 0-starting, protein-wide index to an atom of
  iterest.
-deg: If given angles are given in degree rather than in rad.
-print: For convenience. Prints the atoms and residues from the proteins as an
  ordered list and exits. The name conventions as used by BALL.
-h, --help: Print this help message and exit.

Note that the -print function may be very helpful determining valid names and
indecies.

Contact: Markus-Hermann Koch, markus-hermann.koch@rub.de, August 2013
```

The example call

```
atomGeometry2csv 1A2M.pdb geom.csv -cxxc -res CYS -deg
```

leads to a csv describing the geometry of the atoms included within CYS. In addition extra lines for the cxxc motif are included and all angles are given in degree. A single atom may have more than one line devoted to itself depending on how many bonds of interest to neighbouring atoms it maintains.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	res	bonds	ac	el	x	y	z	a0	el	x	y	z	dist_a0	a1	el	x	y	z	dist_a1	dist_a0_a1	angle	tors_angle	ss_dist
2	CYS	2	CYS_r29_aSG	Sulphur	-28.175	61.408	84.577	CYS-S:CB	Carbon	-28.673	62.098	86.18	1.81486	CYS-S:SG	Sulphur	-29.328	62.445	83.268	2.02935	3.00486	102.683	NA	NA
3	CYS	2	CYS_r32_aSG	Sulphur	-29.328	62.445	83.268	CYS-S:SG	Sulphur	-28.175	61.408	84.577	2.02935	CYS-S:CB	Carbon	-28.318	63.896	82.835	1.82016	3.04059	104.214	90.3266	2.02935
4	CYS	2	r29a0	Nitrogen	-28.077	63.703	87.912	CYS-S:CA	Carbon	-27.549	62.639	87.067	1.45771	PHE:C	Carbon	-27.397	64.214	88.933	1.32889	2.44656	122.731	NA	NA
5	CYS	3	r29a1	Carbon	-27.549	62.639	87.067	CYS-S:N	Nitrogen	-28.077	63.703	87.912	1.45771	CYS-S:C	Carbon	-26.348	63.108	86.243	1.53015	2.47569	111.885	NA	NA
6	CYS	3	r29a1	Carbon	-27.549	62.639	87.067	CYS-S:N	Nitrogen	-28.077	63.703	87.912	1.45771	CYS-S:CB	Carbon	-28.673	62.098	86.18	1.53063	2.43538	109.143	NA	NA
7	CYS	3	r29a1	Carbon	-27.549	62.639	87.067	CYS-S:C	Carbon	-26.348	63.108	86.243	1.53015	CYS-S:CB	Carbon	-28.673	62.098	86.18	1.53063	2.53569	111.879	NA	NA
8	CYS	3	r29a2	Carbon	-26.348	63.108	86.243	CYS-S:CA	Carbon	-27.549	62.639	87.067	1.53015	CYS-S:O	Oxygen	-26.477	63.955	85.358	1.23177	2.40868	121.029	NA	NA
9	CYS	3	r29a2	Carbon	-26.348	63.108	86.243	CYS-S:CA	Carbon	-27.549	62.639	87.067	1.53015	PRO:N	Nitrogen	-25.156	62.556	86.536	1.34589	2.45261	116.885	NA	NA
10	CYS	3	r29a2	Carbon	-26.348	63.108	86.243	CYS-S:O	Oxygen	-26.477	63.955	85.358	1.23177	PRO:N	Nitrogen	-25.156	62.556	86.536	1.34589	2.25609	122.086	NA	NA
11	CYS	1	r29a3	Oxygen	-26.477	63.955	85.358		CYS-S:C	Carbon	-26.348	63.108	86.243	1.23177	NA	NA	NA	NA	NA	NA	NA	NA	NA
12	CYS	2	r29a4	Carbon	-28.673	62.098	86.18	CYS-S:SG	Sulphur	-28.175	61.408	84.577	1.81486	CYS-S:CA	Carbon	-27.549	62.639	87.067	1.53063	2.84734	116.406	NA	NA
13	CYS	2	r29a5	Sulphur	-28.175	61.408	84.577	CYS-S:CB	Carbon	-28.673	62.098	86.18	1.81486	CYS-S:SG	Sulphur	-29.328	62.445	83.268	2.02935	3.00486	102.683	NA	NA
14	CYS	2	r32a0	Nitrogen	-26.328	62.533	82.498	CYS-S:CA	Carbon	-27.18	63.543	81.889	1.45495	HIS:C	Carbon	-25.295	61.989	81.863	1.329	2.44312	122.637	NA	NA
15	CYS	3	r32a1	Carbon	-27.18	63.543	81.889	CYS-S:N	Nitrogen	-26.328	62.533	82.498	1.45495	CYS-S:C	Carbon	-26.391	64.789	81.505	1.52398	2.46568	111.707	NA	NA
16	CYS	3	r32a1	Carbon	-27.18	63.543	81.889	CYS-S:N	Nitrogen	-26.328	62.533	82.498	1.45495	CYS-S:CB	Carbon	-28.318	63.896	82.835	1.52137	2.43545	109.805	NA	NA
17	CYS	3	r32a1	Carbon	-27.18	63.543	81.889	CYS-S:C	Carbon	-26.391	64.789	81.505	1.52398	CYS-S:CB	Carbon	-28.318	63.896	82.835	1.52137	2.50593	110.747	NA	NA
18	CYS	3	r32a2	Carbon	-26.391	64.789	81.505	CYS-S:CA	Carbon	-27.18	63.543	81.889	1.52398	CYS-S:O	Oxygen	-26.675	65.409	80.481	1.23029	2.39153	120.148	NA	NA
19	CYS	3	r32a2	Carbon	-26.391	64.789	81.505	CYS-S:CA	Carbon	-27.18	63.543	81.889	1.52398	TYR:N	Nitrogen	-25.397	65.148	82.318	1.33338	2.43704	116.901	NA	NA
20	CYS	3	r32a2	Carbon	-26.391	64.789	81.505	CYS-S:O	Oxygen	-26.675	65.409	80.481	1.23029	TYR:N	Nitrogen	-25.397	65.148	82.318	1.33338	2.25299	122.949	NA	NA
21	CYS	1	r32a3	Oxygen	-26.675	65.409	80.481		CYS-S:C	Carbon	-26.391	64.789	81.505	1.23029	NA	NA	NA	NA	NA	NA	NA	NA	NA
22	CYS	2	r32a4	Carbon	-28.318	63.896	82.835	CYS-S:SG	Sulphur	-29.328	62.445	83.268	1.82016	CYS-S:CA	Carbon	-27.18	63.543	81.889	1.52137	2.77869	112.211	NA	NA
23	CYS	2	r32a5	Sulphur	-29.328	62.445	83.268	CYS-S:SG	Sulphur	-28.175	61.408	84.577	2.02935	CYS-S:CB	Carbon	-28.318	63.896	82.835	1.82016	3.04059	104.214	90.3266	2.02935

Cysteine bond related geometry data

2.5 movePdb

A simple tool for moving and turning a complete BALL system.

```
Rotates a system around a given axis, then translates it.

Syntax: ./movePdb <input pdb> <output pdb> [<dx>[ <dy>[ <dz>[ rx ry rz angle]]]]

dx,dy,dz: as used in pdb ATOM coordinates in Angstrom.
rx, ry, rz, angle: If one is given all are needed. Rotation axis and angle.
  Angle in degrees.
```

3 C++ classes

3.1 About the Balltools classes

All classes and functions are documented in detail within their header files. This document only intends to give a general overview.

As it is it is still a **TODO** to provide a proper Doxygen[5] documentation as was recommended by Julian.

Only the most important classes will be described here.

3.1.1 Io

A tool class offering functionality for loading and saving pdb files, printing messages to streams and parsing command line parameters for the diverse `balltools`.

3.1.2 MoleculeHandler

A large class for general Molecule modification and analysis. It uses many basic `BALL` functions as well as my own `optimization` library. It offers methods for handling and geometrically measuring secondary structure, iterating through molecules, molecule specific geometric analysis, molecule translation and rotation, molecule structural alignment, bond and hydrogen addition and BALL/Amber energy optimization.

3.1.3 DisulfideHandler

`DisulfideHandler` extends `MoleculeHandler` by methods that are needed to find and modify cxxc cysteine motifs.

Additionally it also offers methods to find and geometrically describe the six form defining secondary structure elements found in oxidoreductase molecules. See `rotationOptimization.ps` for details.

3.1.4 ProteinComparison

At the time of writing (February 2014) this class is still a vague idea. But it will be detailed within the near future. Now that the structural alignment of two related proteins works a usable measure for their alikeness $d(pt_0, pt_1)$ can be implemented.

This will allow for an evaluation of the qualtity of an energy minimization. Consider for instance a molecule with a potential disulfide bond. Within pdb.org there may be two variations: One where the bond is closed and one where it is open. Now, for instance, starting out with the molecule where the bond is open, `disulfideBondMaker` may be used to close the bond and have BALL do an energy optimization given certain paramters. Using a metric like the above-mentioned $d(..)$ the quality of the optimization can be assessed by comparing the freshly created closed-bond molecule to the

[5] http://www.stack.nl/~dimitri/doxygen/

closed-bond molecule from pdb.org. In turn this can be used to improve the parameters for the optimization.

3.2 About the Optimization classes

All classes and functions are documented in detail within their header files.

The optimization classes have their own namespace `optimization`. For matrix and eigen operations as well as singular value decomopsitions they rely on `CBLAS` and `LAPACKe`. So far they comprise of three major classes, and one differentiable function interface with implementations for a rotation around an arbitrary axis with optional translation which is used in `structAlign`.

3.2.1 Matrix

Most of the functions in `CBLAS` and `LAPACKe` are somewhat awkward to use directly. Hence I have decided to implement a simple matrix class that will hide things like row or column majority and offer user-friendly functions like basic Matrix calculation by overloading the arithmetic operators in an intuitive fashion.

There is more convenience stuff like a `toString()` or a `toOctave()` function. The latter will return a string representation that can directly be pasted into an Octave or MATLAB environment.

The other classes in the namespace `optimization` rely on `Matrix`.

3.2.2 LinAlg

Offers everyday task functions like singular value decompositions, Moore Penrose generalized inverse, matrix kernel calculation, Gram-Schmidt orthogonalization, torsion angle, generalized cross product calculations, rank and determinant.

3.2.3 Derivator

This is an abstract class which's implementations describe a function

$$f_\theta(x) : \mathbb{R}^m \mapsto \mathbb{R}^n,\ \theta \in \mathbb{R}^k$$

which is totally differentiable to the second degree. It offers function signatures to get values from up to the second derivate. Such a `Derivator` can be used as an argument for the Newtonizer class.

3.2.4 Newtonizer

The math behind the Newtonizer is explained in great detail within `rotationOptimization.tex`. Basically it takes a function $f_\theta(x)$ which is represented by a suitable implementation of `Derivator`, an initial parameter set θ_0, as well as data sets $\{x_1, \ldots, x_n\}$ and $\{y_1, \ldots, y_n\}$, and a weights n-vector w. Then it attempts to solve the local extreme value problem

$$F_{\theta_{\text{opt}}} = \frac{1}{2} \min_\theta \sum_{j=1}^{n} w_j (f_\theta(x_j) - y_j)^2$$

by finding a null of ∇F_θ.
Here $\theta = (\vartheta_1, \ldots, \vartheta_k)$ and $\nabla = (d/d\vartheta_1, \ldots, d/d\vartheta_k)$.
The Newton iteration starts at θ_0.

3.3 Outlook

For once the `balltools` library will be employed in the metagenomics project for optimization of the parameters used for the energy minimization. To this end a similarity measure for two proteins will be implemented next.

On the other hand especially `structAlign` might offer potential for a publication of its own[6].

[6]Competition, albeit non-secondary structure employing, may be found here: *http://proteopedia.org/wiki/index.php/Structural_alignment_tools*